Campus de l'Université de Montréal
C.P. 6128, succursale Centre-ville
Montréal QC H3C 3J7 Canada

Telephone (514) 343-7575
Telecopieur (514) 343-7121

crt@crt.umontreal.ca
www.crt.umontreal.ca

CRT-2001-40

Centre de recherche sur les transports (C.R.T.)
Centre for Research on Transportation

GENI ANTS FOR THE TRAVELING SALESMAN PROBLEM

by

François-Xavier Lelouarn
Michel Gendreau
Jean-Yves Potvin

October 2001

PUBLICATION
CRT-2001-40

GENI ANTS FOR THE TRAVELING SALESMAN PROBLEM*

by

François-Xavier Lelouarn [2]
Michel Gendreau [1,2]
Jean-Yves Potvin [1,2]

* This research was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC) and by the Fonds pour la Formation de Chercheurs et l'Aide à la Recherche (FCAR).

[1] Centre de recherche sur les transports, Université de Montréal, C.P. 6128, succursale Centre-ville, Montréal, Canada H3C 3J7
[2] Département d'informatique et de recherche opérationnelle, Université de Montréal, C.P. 6128, succursale Centre-ville, Montréal, Canada H3C 3J7

Département d'informatique et de recherche opérationnelle - Publication #1205

Dépôt légal – Bibliothèque nationale du Québec
Bibliothèque nationale du Canada, 2001

Résumé. Dans cet article, l'heuristique probabiliste du plus proche voisin qui est au coeur des systèmes artificiels de colonies de fourmis pour le problème du voyageur de commerce, est remplacée par une procédure d'insertion plus sophistiquée, soit l'heuristique d'insertion GENI. Les performances supérieures des fourmis artificielles GENI sont illustrées sur un ensemble de problèmes tests. Des comparaisons sont établies avec le système classique de colonies de fourmis et une application répétitive de l'algorithme GENI.

Mots clés: systèmes de colonies de fourmis, heuristique d'insertion GENI, problème du voyageur de commerce.

Abstract. In this paper, the probabilistic nearest neighbor heuristic, which is at the core of classical ant colony systems for the Traveling Salesman Problem, is replaced by an alternative, more complex, insertion procedure known as the GENI heuristic. The benefits provided by GENI-based ants are empirically demonstrated on a set of benchmark problems, through a comparison with the classical ant colony system and an iterated GENI heuristic.

Keywords: ant colony systems, GENI insertion heuristic, Traveling Salesman Problem.

1 Introduction

Ant systems represent a relatively novel heuristic approach for solving combinatorial optimization problems [7]. Its motivation comes from the way real ants find short paths from their nest to food sources by laying down pheromone traces on the ground. Although considered for a variety of combinatorial problems since their introduction in the early 90's [1, 2, 8], ant systems have been applied first on the classical Traveling Salesman Problem (TSP) [3, 4]. This did not come as a surprise because the TSP, although NP-hard, is a canonical problem which can be very easily stated: given a complete graph $G = (V, E)$, where V is the set of n vertices or cities and E is the set of edges, and a traveling distance between each pair of vertices, the problem is to find the shortest tour that visits each vertex exactly once [10]. In this paper, we focus on Euclidean TSPs, where the distance between vertices is defined through the Euclidean metric.

Ant systems for the TSP construct solutions by iteratively adding new cities to the current partial tour. The choice of the next city to visit is based on a greedy heuristic, typically the nearest neighbor heuristic, which considers both the distance to the next city and the amount of pheromone deposited on the link to that city. The latter represents the "memory" of the system as it indicates how often that link was used in the past to find good solutions. In this paper, we propose the alternative, more complex, GENI (GENeralized Insertion) heuristic [9] for incrementally extending the partial tour, and we investigate the impact on solution quality of these more sophisticated GENI-based ants.

The rest of the paper is organized as follows. Section 2 first briefly introduces the principles of ant systems, upon which a more recent metaphor, known as the Ant Colony System (ACS), is derived [5]. Section 3 then describes the GENI insertion method for the TSP [9]. Section 4 explains how GENI was integrated within the ACS framework. Computational results on Euclidean TSPs are reported in Section 5. Finally, concluding remarks are made.

2 Ant Colony System for the TSP

When ants look for food sources, they initially look around in a random manner. When they find something, they come back to the nest, laying down a chemical compound on the ground, known as pheromone, on their way back. The quantity of pheromone is related to the quality of the food source, namely the quantity of food and the distance from the nest. The next ants will thus search in a less random fashion, as they will be attracted by the pheromone traces. More ants will be attracted on paths with more pheromone traces, which in turn will lead to more and more pheromone laid down on these paths. Ultimately, all ants will be attracted by the more interesting paths. Ant systems are motivated by this way of communicating through pheromone traces to find good paths.

In the following, the recent ACS framework for solving the TSP is presented [5] (readers interested in previous ant-based metaphors are referred to [3, 4, 6]). Broadly speaking, the ACS works as follows: m ants are positioned on the n cities, according to some a priori assignment procedure (which may be a random assignment). At each iteration of the algorithm, each ant builds a tour by repeatedly selecting a new city to visit through the application of a probabilistic nearest neighbor heuristic. While constructing a tour, the ant also modifies the amount of pheromone on the edges it follows by applying a local update rule. When all ants have completed their respective tours, the amount of pheromone on each edge is modified again through a global update rule which favors the edges associated with the best tour.

During the solution construction process, the ants are guided both by the amount of pheromone on the edges and by the length of these edges. In particular, a short edge with a high amount of pheromone is highly desirable, since it was part of many good solutions in the past and it only slightly increases the total length of the tour.

The ACS procedure is summarized below using a pseudo-code notation. The main steps of this algorithm are then explained.

1. randomly assign each ant to a starting city;
2. put some a priori level of pheromone on every edge;
3. for I iterations do:
   - 3.1 for each ant, construct a tour and update the pheromone locally;
   - 3.2 update the pheromone globally;
4. output the best solution found.

2.1 State transition rule

In Step 3.1, the ant constructs a tour through the repeated addition of new cities. Assuming that ant r is currently located at city i, a state transition rule is used to select the next city j to move to, namely:

$$j = \begin{cases} \text{argmax}_{k \in J_r}(\tau_{ik}\eta_{ik}^{\beta}), & \text{if } q \leq q_0 \\ J, & \text{otherwise} \end{cases} \tag{1}$$

where J is a random variable with the probability distribution

$$p(J = j) = \begin{cases} \frac{\tau_{ij}\eta_{ij}^{\beta}}{\sum_{k \in J_r} \tau_{ik}\eta_{ik}^{\beta}} & \text{if } j \in J_r \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

and where η_{ij} is the inverse of the length c_{ij} of edge (i,j), τ_{ij} is the amount of pheromone on edge (i,j), β is a weighting parameter that determines the relative importance of length versus pheromone, q is a random number uniformly distributed between 0 and 1, q_0 is a threshold parameter and J_r is the set of cities not yet visited by ant r.

From these equations, we can see that edge (i,j) looks better when its length is shorter and the amount of pheromone is higher. Parameter $\beta > 0$ is used to put more or less emphasis on length versus pheromone. When a randomly generated value q is less than or equal to the threshold q_0, the best edge according to equation 1 is chosen to reach the next city. Otherwise, equation 2 is used to probabilistically select the next city to be visited, with a bias towards the best edges (namely, the selection probability is proportional to edge quality).

2.2 Local update rule

When ant r constructs a tour, the pheromone level on each edge (i,j) visited by the ant is updated as follows:

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij} + \rho\tau_0 \tag{3}$$

where $0 < \rho < 1$ is a parameter and τ_0 is the initial level of pheromone [5]. When ρ is high, the amount of pheromone is substantially reduced on the edges of the tour constructed by the ant. Consequently, the other ants will less likely follow the same tour, and the search will be more diversified. This could be beneficial, although care must be taken not to alter too much the pheromone traces to avoid producing a random search pattern.

2.3 Global update rule

The global update rule is aimed at favoring the best tour constructed by the ants at the current iteration by putting more pheromone on its edges. It thus leads to an intensification of the search around that solution. At the same time, some amount of pheromone is removed from each edge (to mimic the "evaporation" of pheromone observed in nature). This is expressed by:

$$\tau_{ij} \leftarrow (1-\alpha)\tau_{ij} + \alpha\Delta\tau_{ij} \tag{4}$$

where

$$\Delta\tau_{ij} = \begin{cases} (L^*)^{-1}, & \text{if } (i,j) \text{ is in the best tour} \\ 0 & \text{otherwise} \end{cases}$$

and where $0 < \alpha < 1$ and L^* is the length of the best tour. For computational efficiency, we have divided the right-hand side of (4) by $1-\alpha$, to obtain (5).

Given the definition of $\Delta\tau_{ij}$, the pheromone levels now need to be adjusted only on the edges of the best tour, with is of complexity $O(n)$ rather than $O(n^2)$.

$$\tau_{ij} \leftarrow \tau_{ij} + \frac{\alpha}{1-\alpha}\Delta\tau_{ij} \tag{5}$$

This ACS framework will be used in the following, although the more powerful GENI insertion heuristic will be applied in place of the nearest neighbor heuristic to construct solutions. In the next section, GENI is introduced.

3 The GENI heuristic

GENI [9] is an iterative insertion heuristic, where each new vertex u is inserted between two vertices i and j that are not necessarily consecutive in the current tour. However, after the insertion, both i and j will be adjacent to u. Let assume, for a clockwise orientation of the tour, that vertex k is on the path from j to i, and vertex l is on the path from i to j. Let also assume that for any given vertex h, $h-1$ and $h+1$ are the predecessor and successor, respectively, of h.

Then, two types of insertions are considered (see Figure 1). In type I, for $k \neq i, j$, the insertion of vertex u in the tour leads to the deletion of edges $(i, i+1)$, $(j, j+1)$ and $(k, k+1)$ and their replacement by (i, u), (u, j), $(i+1, k)$ and $(j+1, k+1)$. In type II, for $l \neq i, i+1$ and $k \neq j, j+1$, the insertion of vertex u in the tour leads to the deletion of edges $(i, i+1)$, $(l-1, l)$, $(j, j+1)$ and $(k, k-1)$ and their replacement by (i, u), (u, j), $(l, j+1)$, $(k-1, l-1)$ and $(i+1, k)$.

These two types of insertions are also considered for a counter-clockwise orientation of the tour, for a total of four different types of insertions. Since the number of ways to choose the four vertices i,j,k and l for a given type of insertion is of complexity $O(n^4)$, GENI only considers vertices that are close enough, through the definition of the so-called p-neighborhood of each vertex. Namely, the p-neighborhood of vertex u, noted $N_p(u)$, is the set of p vertices on the tour that are the closest to u. Then, for a given choice of parameter p, GENI only looks for insertions such that $i, j \in N_p(u)$, $k \in N_p(i+1)$ and $l \in N_p(j+1)$. In addition, the insertion of u between two consecutive vertices i and $i+1$ is examined if $i \in N_p(u)$. In practice, p is set to a relatively small value, usually between 4 and 7.

The GENI algorithm may now be summarized as follows:

1. create an initial tour by randomly selecting a subset of three vertices;
2. initialize the p-neighborhood of each vertex;
3. randomly select a vertex u not yet included in the tour;

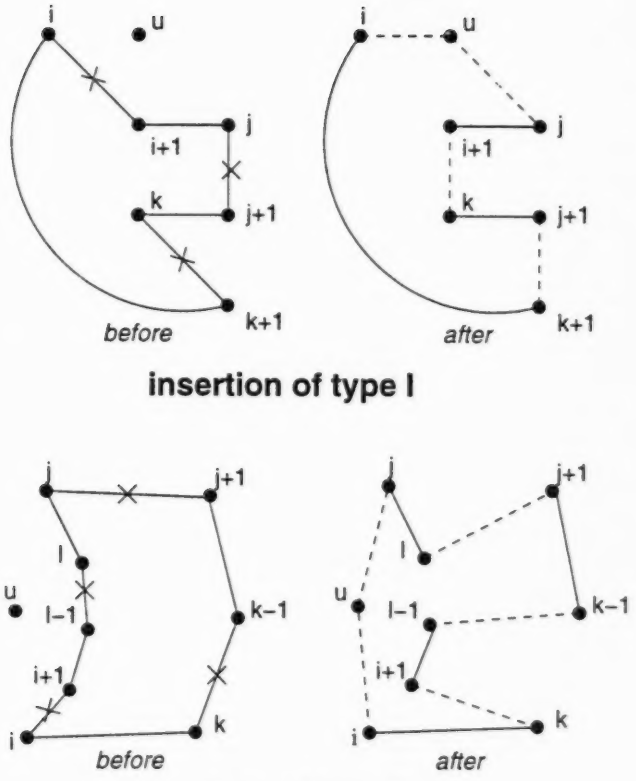

Figure 1: GENI insertions

4. insert it at least cost, by considering all insertions of type I and II;

5. update the p-neighborhood of each vertex (given that u is now in the tour);

6. if all vertices are in the tour, STOP. Otherwise, go back to 3.

4 The GACS

This section describes how the GENI heuristic presented in section 3, has been integrated within the ACS framework to produce the GACS (GENI Ant Colony System). This integration leads to two important modifications to GENI which impact the evaluation and selection of the next move to be performed: (1) the cost of an edge now depends on its length and the amount of pheromone deposited on it and (2) a probabilistic state transition rule is used to choose the next insertion to be performed (c.f., step 4 of the GENI algorithm). Since the local and global update of pheromone traces remain unchanged with regard to the original ACS, we focus on these two issues in the following.

4.1 Adjusted length

In the ACS framework, the amount of pheromone deposited on the edges is the mechanism by which the ants communicate together to share information about good paths (good solutions). These pheromone traces can thus be regarded as the memory of the system. In the classical ACS, this is taken into account in the probabilistic state transition rule, where the quality of edge (i,j) is expressed as the ratio of τ_{ij} on c_{ij}^{β} (see equations 1 and 2). Similarly, in the context of the GENI insertion heuristic, the length of each edge (i,j) is modified by the amount of pheromone deposited on that edge. The formula for the adjusted length is:

$$c'_{ij} = \frac{c_{ij}}{1 + \beta \cdot \tau_{ij}^{R}} \tag{6}$$

where τ_{ij}^{R} is the relative amount of pheromone on edge (i,j). Namely:

$$\tau_{ij}^{R} = \frac{\tau_{ij}}{\max_{(k,l)\in E}(\tau_{kl})} \tag{7}$$

In this equation, the amount of pheromone on each edge is normalized between 0 and 1. If the maximum in equation 7 is equal to 0 (i.e., there is no pheromone on the edges of the graph), τ_{ij}^{R} is set to 0 for every edge. As in the original ACS, the relative importance of the pheromone is adjusted through parameter β.

4.2 State transition rule

Consider the state transition rule of the classical ACS where some ant r, currently located at city i, must choose which city to visit next among those that are yet unvisited. In that case, city j is selected according to a probabilistic state transition rule, where the selection probability of city j is proportional to the ratio of τ_{ij} on c_{ij}^{β}. Then, city j is inserted at the end of the sequence. Clearly, the solution produced would always be the same, for a given starting city, without the probabilistic bias. The latter, by modifying the selection order of the cities, allows the method to generate different solution

Since the insertion mechanism at the core of GENI is more complex, it is less sensitive to the selection order of the cities. As opposed to the classical ACS, where a different selection order necessarily leads to a different solution (since each selected city is simply inserted at the end of the solution), GENI may well end up with two identical solutions, even for two different selection orders. An additional mechanism is thus provided to diversify the search. This is done by using a probabilistic state transition rule similar to the one found in the ACS, to determine what insertion will be performed on the randomly chosen city (instead of always performing the best insertion). The number of transitions from the current solution thus corresponds to the number of insertions of type I and II that may be applied to the randomly chosen city. This number is typically much larger than in the case of the ACS (where the number of transitions is equal to the number of cities that remain to be visited).

To associate a selection probability with each GENI insertion, a rank-based approach is used in place of the pure proportional scheme [12]. This approach does not put any emphasis on the magnitude of the difference between two competing insertions to derive the probability distribution. Only their relative order is important. With this approach, two insertions of almost equal value will be better discriminated, while a dominant insertion (i.e., an insertion that is much better than the others) will not be assigned a selection probability too close to 1, to avoid a too strongly deterministic search pattern.

The rank-based scheme works as follows for a given randomly selected city. First, the S best GENI insertions, over all insertions of types I and II, are considered and ranked from the one that leads to the smallest increase to the tour length (rank S) to the one that leads to the largest increase (rank 1). Preliminary tests have shown that parameter S is important and should be set to a small value, like S=5 in our experiments, to avoid the selection of a bad insertion that could be difficult to "repair" in the following iterations. The state transition rule then consists in selecting a rank t between 1 and S, and applying the insertion with the corresponding rank, that is:

$$t = \begin{cases} S, & \text{if } q \leq q_0 \\ T, & \text{otherwise} \end{cases} \tag{8}$$

where T is a random variable with the probability distribution

$$p(T = t) = \frac{t}{\sum_{t'=1}^{S} t'} = \frac{2t}{S(S+1)}, \; t = 1, ..., S. \tag{9}$$

Clearly, the insertion of rank $t = S$, which leads to the smallest increase to the tour length, has the highest selection probability, according to equation 9. Conversely, the insertion of rank $t = 1$, which leads to the largest increase, has the lowest selection probability. The probabilities for the other insertions are linearly distributed between these two values, with a constant gap between two insertions with consecutive ranks.

4.3 The algorithm

The GACS can now be summarized as follows:

1. initialize the pheromone level on each edge;
2. for I iterations do:
   - 2.1 for each ant do:
     - 2.1.1 create an initial tour by randomly selecting three vertices;
     - 2.1.2 initialize the p-neighborhood of each vertex;
     - 2.1.3 repeat until a complete tour is obtained:
       - randomly select a vertex u not yet included in the tour;
       - select and apply a GENI insertion to vertex u using the state transition rule;
       - update the p-neighborhood of each vertex;
     - 2.1.4 update pheromone locally;
   - 2.2 update pheromone globally;
3. output best solution found.

Replacing the nearest neighbor heuristic by the GENI heuristic increases the complexity from $O(Imn^2)$ to $O(Im(np^4 + n^2))$, where m is the number of ants and n is the number of cities. Each iteration of the modified procedure is thus more computationally expensive, but is expected to lead to larger improvements. In the next section, computational results on Euclidean TSPs are reported and a comparison with the original ACS is provided.

5 Computational Results

The GACS was implemented in C and the tests were performed on a 400MHz UltraSparc2 processor. We first examine the impact of different parameter settings on solution quality. Then, a comparison with the multi-start GENI heuristic, namely the GACS without pheromone traces, and the ACS is presented on the five TSPLIB [11] library problems used by Dorigo et al. in [5].

5.1 Parameter sensitivity

To study the impact of different parameter values, six TSP instances have been used. Three instances, *alea100*, *alea300* and *alea500* of size 100, 300 and 500 respectively, were generated randomly. The three other instances, *kroA100*, *pr299* and *u574* of size 100, 299 and 574, respectively, were taken from the TSPLIB library. The idea was to calibrate the algorithm on instances that are different from those used for comparison with the ACS. Each result reported is the average of 10 different runs, using a number of iterations I equal to either n/m or $10n/m$. We used m=10 ants in all experiments (a theoretical justification for this number may be found in [5]). Also, the initial amount of pheromone τ_0 was set to 0. Consequently the adjusted length corresponds to the original length at the start of the algorithm (see equation 6).

When the value of one parameter was tested, the others were set at their default value, namely: ρ=0.5, α=0.5, β=0.5 and q_0=0.95. The value for the neighborhood parameter in GENI was fixed at p=7.

Parameter β

Table 1 shows the impact of parameter β on solution quality. Through this parameter, it is possible to adjust the relative importance of pheromone traces when evaluating the cost of an edge. These results indicate that incorporating pheromone into the edge cost has generally a positive impact on solution quality. The values leading to the best solutions vary widely between 0 and 1, but a larger number of best results are obtained with either $\beta = 0.25$ or $\beta = 0.5$.

Parameter α

Table 2 shows the impact of parameter α on solution quality. This parameter is used in the global update rule, when the pheromone is added on the edges of the best tour. For $\alpha = 1$, equation 4 was applied rather than equation 5, to avoid the division by 0. Clearly, the best solutions are found with smaller values of α, like $\alpha = 0.1$ or $\alpha = 0.25$. This makes sense, as larger values tend to strongly bias the search toward elite solutions and to lead to premature convergence.

Parameter ρ

Table 3 shows the impact of parameter ρ on solution quality. This parameter is used in the local update rule, when some pheromone is removed from the edges

Table 1: Impact of parameter β on solution quality

Problem	Iter.	β					
		0	0.25	0.5	0.75	1	5
kroA100	n/m	21361.3	21339.6	**21283.0**	21365.2	21357.5	21354.9
	$10n/m$	21288.3	**21283.0**	**21283.0**	**21283.0**	21286.6	21284.6
pr299	n/m	49228.2	**49184.3**	49217.1	49258.4	49308.7	49581.1
	$10n/m$	48895.9	48915.7	48850.1	48918.9	**48848.7**	49049.3
u574	n/m	38620.0	38611.7	**38606.0**	38651.8	38696.7	38838.9
	$10n/m$	**38362.0**	38416.0	38393.2	38398.6	38368.0	38542.9
alea100	n/m	794.3	**792.4**	**792.4**	794.3	794.3	794.3
	$10n/m$	785.4	784.3	784.3	783.6	**783.0**	784.3
alea300	n/m	**4021.6**	4022.2	**4021.6**	4029.3	4024.1	4031.6
	$10n/m$	3998.7	**3990.1**	3996.5	3994.2	3998.2	4004.9
alea500	n/m	8640.2	8623.8	8636.7	**8622.7**	8651.5	8686.5
	$10n/m$	8593.6	**8572.2**	8582.6	8578.1	8573.6	8635.5

Table 2: Impact of parameter α on solution quality

Problem	Iter.	α					
		0	0.1	0.25	0.5	0.75	1
kroA100	n/m	21361.3	**21336.8**	**21336.8**	**21336.8**	**21336.8**	21548.2
	$10n/m$	21288.3	**21286.6**	**21286.6**	**21286.6**	**21286.6**	21548.2
pr299	n/m	49228.2	**49133.3**	49197.9	49214.1	49214.0	50033.4
	$10n/m$	48895.9	48868.1	48872.8	48861.8	**48841.9**	50033.4
u574	n/m	38620.0	38578.4	**38564.4**	38606.0	38609.6	39159.7
	$10n/m$	38362.0	38389.8	**38364.9**	38393.2	38397.0	39159.7
alea100	n/m	794.3	**790.2**	**790.2**	**790.2**	**790.2**	803.1
	$10n/m$	785.4	**784.3**	**784.3**	**784.3**	**784.3**	803.1
alea300	n/m	4021.6	**4016.4**	4018.7	4021.8	4019.4	4091.3
	$10n/m$	3998.7	**3985.0**	3997.6	3996.5	3997.3	4091.3
alea500	n/m	8640.2	8647.1	**8636.3**	**8636.3**	8650.6	8751.8
	$10n/m$	8593.6	8584.1	**8576.3**	8582.6	8577.8	8751.8

Table 3: Impact of parameter ρ on solution quality

Problem	Iter.	ρ					
		0	0.1	0.25	0.5	0.75	1
kroA100	n/m	21393.1	21381.6	21346.8	**21332.5**	21359.5	21354.9
	$10n/m$	21294.0	21292.7	21287.2	**21283.0**	21284.3	21286.6
pr299	n/m	49345.3	49330.7	49218.4	49214.1	**49151.3**	49246.7
	$10n/m$	48863.9	48871.6	48859.4	48861.8	**48853.6**	48871.7
u574	n/m	38898.4	38590.0	38675.2	38606.0	38611.3	**38534.5**
	$10n/m$	38503.0	38402.6	38449.0	38393.2	38399.0	**38318.9**
alea100	n/m	795.7	795.7	**791.9**	792.4	792.4	795.2
	$10n/m$	785.9	**783.0**	784.3	784.3	784.3	785.3
alea300	n/m	4057.0	4018.0	4017.4	4021.8	**4013.1**	4027.6
	$10n/m$	4005.2	**3988.5**	3995.9	3996.5	3990.7	4000.2
alea500	n/m	8696.6	8646.1	8655.0	8636.7	**8618.2**	8655.5
	$10n/m$	8608.2	8590.6	**8565.5**	8582.6	8578.9	8594.9

of the current tour to diversify the search. Clearly, the best values are found with larger values of ρ, like $\rho = 0.75$, which indicates that a significant level of diversification is desirable.

Parameter q_0

Table 4 shows the impact of parameter q_0 on solution quality. This parameter is used in equation 8 when the GENI insertion, to be applied to the currently selected city, is chosen. When $q_0 = 1$, the best GENI insertion is always chosen. When $q_0 < 1$, some perturbation is introduced, using a probability distribution biased toward the best insertions. The numbers in the Table indicate that values closer to 1 are better, although a slight amount of perturbation is desirable.

5.2 Comparison with ACS and a multi-start GENI

The ants communicate together through pheromone traces, which contain information about the past history of the search. By removing those traces, we obtain a multi-start GENI heuristic (mGENI), where each run is performed independently of the previous ones. This section provides a comparison of GACS with this algorithm and the classical ACS on the same five problems used by Dorigo et al. in [5]. The three methods were given the same amount of computation time on each instance, based on the time reported in [5], which is the time to reach the best solution with ACS. This value was then divided by 9, given that our processor is nine times faster than the one used by Dorigo et al.

Table 4: Impact of parameter q_0 on solution quality

Problem	Iter.	q_0			
		0.9	0.95	0.98	1
kroA100	n/m	21355.5	21314.5	21293.5	**21291.7**
	$10n/m$	21302.0	21283.0	**21282.0**	**21282.0**
pr299	n/m	49764.9	48980.1	**48842.3**	48884.6
	$10n/m$	49183.4	48701.1	**48571.1**	48629.6
u574	n/m	38959.2	38377.9	**38261.6**	38277.4
	$10n/m$	38724.6	38196.7	**38109.8**	38139.8
alea100	n/m	795.5	790.0	**789.6**	790.6
	$10n/m$	786.9	781.5	781.3	**780.6**
alea300	n/m	4047.8	4001.0	4001.6	**3999.5**
	$10n/m$	4024.5	3970.3	**3961.4**	3965.6
alea500	n/m	8723.5	8568.7	**8548.4**	8550.0
	$10n/m$	8657.3	**8514.7**	8526.4	8519.7

For the GACS, the parameter values were set to: $m = 10$, $\beta = 0.5$, $\alpha = 0.25$, $\rho = 0.75$ and $q_0 = 0.98$.

The first column in Table 5 corresponds to the problem identifier. The second column is the optimum value for the corresponding problem. The third column CPU is the computation time in seconds allocated to each method to solve each problem. Under GACS, mGENI and ACS, we find the length of the best overall solution after 15 different runs, the average length over all runs, the average % over the optimum over all runs and the average time to reach the best solution over all runs, in this order. Note that only GACS and mGENI were implemented on our machine; the results for ACS were taken from [5].

We can see that GACS provides better solutions than ACS on problems d198, att532 and fl1577. It is outperformed on problem pcb442 and, to a much lesser extent, on problem rat783. The average solutions produced by GACS on the five benchmark problems are between 0.24% and 3.28% over the optimum, with an average of 1.94%, while those produced by ACS are between 1.74% and 4.11% over the optimum, with an average of 2.55%. GACS also converges faster to its best solutions. The results also indicate that GACS is more stable and exhibits less variance with regard to solution quality from one run to another. Finally, the comparison between GACS and mGENI illustrates the benefits provided by the communication scheme among the ants, from one iteration to the next, through pheromone traces. The multistart GENI stands between ACS and GACS with regard to the average solution quality, at 2.24% above the optimum.

Table 5: Comparison between GACS, mGENI and ACS

Problem	Optimum	CPU	GACS	mGENI	ACS
d198	15780	1300s	15795	15811	15888
			15818	15885	16054
			0.24%	0.29%	1.74%
			823s	897s	1300s
pcb442	50778	3305s	51727	51749	51268
			51941	51996	51690
			2.29%	2.40%	1.80%
			1750s	1700s	3305s
att532	27686	6460s	28052	28082	28147
			28148	28271	28523
			1.67%	2.00%	3.02%
			3962s	5033s	6460s
rat783	8806	14320s	9072	9066	9015
			9095	9095	9066
			3.28%	3.28%	2.95%
			8138s	6798s	14320s
fl1577	22249	50240s	22599	22684	22977
			22722	22862	23163
			2.13%	2.26%	4.11%
			31725s	34384	50240s
Avg.	25060	15125s	25449	25478	25459
			25545	25622	25699
			1.94%	2.24%	2.55%
			9280s	9762s	15125s

6 Conclusion

This paper has proposed the use of more sophisticated GENI-based ants within the ACS framework. The benefits of this approach have been illustrated on a set of benchmark problems. Clearly, our intent was not to generate the best possible solutions. If so, additional mechanisms could have been incorporated either at the end of our algorithm or after each iteration, like local search procedures based on edge exchanges. By doing this, however, the benefits provided by GENI ants could hardly be isolated.

Acknowledgments. Financial support for this work was provided by the Canadian Natural Sciences and Engineering Research Council (NSERC) and by the Quebec Fonds pour la Formation de Chercheurs et l'Aide à la Recherche (FCAR). This support is gratefully acknowledged. Thanks also to Jean-Philippe Tardif for running some computational experiments

References

[1] Bullnheimer B., Hartl R.F., Strauss C., "Applying the Ant System to the Vehicle Routing Problem", in *Meta-heuristics - Advances and Trends in Local Search Paradigms for Optimization*, S. Voss, S. Martello, I.H. Osman and C. Roucairol eds., Kluwer, 285–296, 1999.

[2] Colorni A., Dorigo M., Maniezzo V., Trubian M., "Ant System for Job-Shop Scheduling", *Belgian Journal of Operations Research, Statistics and Computer Science 34*, 39–53, 1994.

[3] Colorni A., Dorigo M., Maniezzo V., "An Investigation of Some Properties of an Ant Algorithm" in *Parallel Problem-Solving from Nature 2*, R. Männer and B. Manderick eds., North-Holland, 509–520, 1992.

[4] Colorni A., Dorigo M., Maniezzo V., "Distributed Optimization by Ant Colonies", in *Proceedings of the European Conference on Artificial Life*, Paris, France, 134–142, 1991.

[5] Dorigo M., Gambardella L.M., "Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem", *IEEE Trans. on Evolutionary Computation 1*, 53–66, 1997.

[6] Dorigo M., Maniezzo V., Colorni A., "Ant System: Optimization by a Colony of Cooperating Agents", *IEEE Trans. on Systems, Man and Cybernetics 26*, 29–41, 1996.

[7] Dorigo M., "Optimization, Learning and Natural Algorithms", Ph.D. Dissertation, DEI, Politecnico di Milano, Italy, 1992 (in italian).

[8] Gambardella L.-M., Taillard E.D., Dorigo M., "Ant Colonies for the Quadratic Assignment Problem", *Journal of the Operational Research Society 50*, 167–176, 1999.

[9] Gendreau M., Hertz A., Laporte G., "New Insertion and Postoptimization Procedures for the Traveling Salesman Problem", *Operations Research 40*, 1086–1094, 1992.

[10] Lawler E.L., Lenstra J.K., Rinnooy Kan A.H.G., Schmoys D.B. eds., The Traveling Salesman Problem, Wiley, 1985.

[11] Reinelt G., "TSPLIB - A Traveling Salesman Problem Library", *ORSA Journal on Computing 3*, 376–384, 1991.

[12] Whitley D., "The GENITOR Algorithm: Why Rank-Based Allocation of Reproductive Trials is Best", in *Proceedings of the Third Int. Conf. on Genetic Algorithms*, J.D. Schaffer ed., Morgan Kaufmann, 116–121, 1989.